TR52: final examination

Spring semester - 2010

1 Questions on the Ada language

Please, answer to the following questions (please no more than 5 lines each answer):

1. What is the basic element or construction for encapsulation in Ada? (Reminder: you can "encapsulate" everywhere.)
2. Mention some programmer's interests or advantages in using Ada packages.
3. In the case of Ada's generic procedures and generic packages, what kind of attributes can be generic parameters ? (Mention at least one kind of attributes).
4. Mention the two basic mechanisms for handling exceptions. Which one is proposed in Ada ? Which of these two mechanisms is more similar to the preemption mechanism ?
5. How an Ada task puts itself to wait during a fixed period of time ?
6. Describe briefly the watchdog mechanism of the Ada language.
7. Consider the following source code :

```
WITH Ada.Text_IO;
WITH Ada.Integer_Text_IO;
PROCEDURE task_demo_01 IS
```

```
        TASK TYPE intro_task (message : Integer);
        TASK BODY intro_task IS
        BEGIN
                  Ada.Text_IO.put (Item => "From Task ");
                  Ada.Integer_Text_IO.put (Item => message, Width => 1);
                  Ada.Text_IO.new_line;
        END intro_task;

        Task_1 : intro_task (message => 1);

        Task_2 : intro_task (message => 2);

        Task_3 : intro_task (message => 3);

BEGIN
NULL;
END task_demo_01;
```

When the program is executed many times, messages can be displayed in different orders at each execution :

```
From Task 3        From Task 1        From Task 3
From Task 1   or   From Task 3   or   From Task 2    ...
From Task 2        From Task 2        From Task 1
```

Explain how to use the rendez-vous mechanism of Ada to obtain the following output, at all executions:

```
From Task 1
From Task 2
From Task 3
```

2 Exercice on Real Time Java

A traffic regulation system uses several cameras to monitor the flow of vehicles at a crossroad. The cameras are placed on top of the traffic lights of

each street that leads to the crossroad. Each camera points to the queue of cars that are waiting at its corresponding traffic light. The traffic regulation system is built on a real time java computer and is designed with two layers.

The first layer is composed of several threads (class **QueueAnalyzer**) that have to estimate the number of waiting cars at the traffic lights. There is one such thread linked to each camera of the crossroad. The logic of the class **QueueAnalyzer** is to capture an image (method "**TrImage getImg()**") and to analyze it in order to determine the number of waiting cars (method "**int getCarNbr(TrImage)**"). The class **TrImage** is a given class that represents the images gotten from the cameras. These threads will be periodic with a 1 sec period, and a estimation cost of 15ms.

The second layer holds a single thread that is the traffic regulator (class **TrafficRegul**). Its goal is to choose the traffic light that has the most vehicles waiting, and turn this traffic light green. The class **TrafficRegul** uses the following given methods:

- method "**int selectTrafficLight(int[] tabCarNbr)**" will determine which traffic light should be turned green. The parameter is an int array that gives the number of waiting cars, $tabCarNbr[i]$, for the traffic light of indice i.
- method "**void setGreen(int tl)**" turns the traffic light of indice tl to green. It allows thus the corresponding waiting cars to drive.
- method "**void setRed(int tl)**" turns the traffic light of indice tl to red.

The class **TrafficRegul** has to be a periodic thread with a period of 1 sec, and an estimation cost of 30 ms.

1. Propose a realtime solution principle for this problem, and write an UML diagram class.
2. Give the constructor and the logic of the two main realtime java threads (class **QueueAnalyzer** and class **TrafficRegul**), as well as other classes that you may need for this traffic regulation system.
3. The proposed system has a major drawback, when one street has a big traffic, and another as little waiting vehicles. In this case, these cars will wait too long to drive through the crossroad. Propose a real time

solution (only the design idea) in order that every car do not have to wait more than five minutes.

3 Exercice on the UPPAAL modeling language

Figure 1 presents the end of an assembly line, where parts are transferred from TSA to B1 by RA1 or from TSA to B2 by RA2. Parts arrive to the TSA every t_a seconds, with $t_a \in \{5, 6, 7\}$ (with equal probability). Transfer time from TSA to B1 by RA1 or to B2 by RA2 equals 6 seconds (total cycle time). A failure of RA1 is detected if RA1 does not send a signal EOT (end of transfer) 10 seconds after the beginning of a transfer. In this case, RA2 is activated, and begins to transfer parts to B2.

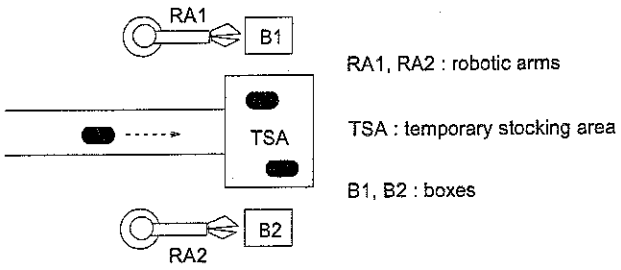

Figure 1: Assembly line termination

Work to be done :

1. Model this system with the UPPAAL automata language (include declarations). Modularity will be appreciated (for instance: a supervisor automaton, robot automata, ...)

2. Express, using the UPPAAL property language, the following properties:

   P1: The number of pieces in the TSA always remain < 5.

   P2: If a failure of RA1 is detected, pieces begin to arrive to box B2.